REJ10J1253-0300

Everywhere you imagine. RENESAS

E8 Emulator

Additional Document for User's Manual
R0E000080KCE00EP4

Renesas Microcomputer Development Environment System
M16C Family / M16C/60 Series
Notes on Connecting the M16C/62P, M16C/6N4, M16C/6N5, M16C/6NK, M16C/6NM, M16C/6NL and M16C/6NN

User's Manual

Rev.3.00
November 01, 2006

Renesas Technology
www.renesas.com

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of non-flammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/ or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.

Contents

This user’s manual is applicable to the E8 emulator software V.2.09 Release 00 or later.

Section 1 Specifications of the E8 Emulator

Table 1.1 shows the specifications of the M16C/62P and M16C/6N Groups E8 Emulator.
This manual describes the M16C/6N4, M16C/6N5, M16C/6NK, M16C/6NM, M16C/6NL, M16C/6NN groups as the M16C/6N group.

Table 1.1 Specifications of the M16C/62P and M16C/6N Groups E8 Emulator

Target MCU	M16C Family M16C/60 Series M16C/62P and M16C/6N Groups	
Usable operating mode	Single-chip mode, Memory expansion mode * Microprocessor mode is not supported.	
Break function	- Address match break, 8 points - PC break (up to 255 points) - Forcible break	
Trace function	Not available	
Flash memory programming function	Available	
User interface	Clock-synchronized serial (communicating via P64/P65/P66/P67)	
MCU resource to be used	- ROM size: 2 KB (alterable assigned address) - RAM size: 128 bytes (alterable assigned address) - Stack 14 bytes - UART 1 function and P64/P65/P66/P67 - Pins P50 and P55 - Address match interrupt	
Emulator power supply	Unnecessary (USB bus powered, power supplied from the PC)	
Interface with host machine	USB (USB 1.1, full speed) * Also connectable to host computers that support USB 2.0	
Power supply function	Can supply 3.3 V or 5.0 V to the target board (300mA, max)	
Power voltage	M16C/62P	3.0--3.6V, 4.5--5.5V
	M16C/6N4 (Normal-version) M16C/6N5 (Normal-version) M16C/6NK (Normal-version) M16C/6NL M16C/6NN	3.0--3.6V, 4.5--5.5V
	M16C/6N4 (T-version, V-version) M16C/6N5 (T-version, V-version) M16C/6NK (T-version, V-version) M16C/6NM (T-version, V-version)	4.5--5.5V

Table 1.2 shows the operating environment of the E8 Emulator.

Table 1.2 Operating Environment

Temperature	Operating	: 10°C to 35°C
	Storage	: –10°C to 50°C
Humidity	Operating	: 35% RH to 80% RH, no condensation
	Storage	: 35% RH to 80% RH, no condensation
Vibration	Operating	: 2.45 m/s^2 max.
	Storage	: 4.9 m/s^2 max.
	Transportation	: 14.7 m/s^2 max.
Ambient gas	No corrosive gas	

Section 2 Connecting the Emulator with the User System

Before connecting an E8 emulator with the user system, a connector must be installed in the user system so that a user system interface cable can be connected. When designing the user system, refer to Figure 3.1, Pin Assignments of the E8 Connector, and Figure 4.1, Example of E8 Connection, shown in this manual.
Before designing the user system, be sure to read the E8 emulator user's manual and the hardware manual for related MCUs.

Table 2.1 shows the recommended connector for the emulator.

Table 2.1 Recommended Connector

	Type Number	Manufacturer	Specifications
14-pin connector	7614-6002OO*	3M Limited	14-pin straight type

*OO indicates coat specification

Connect pins 2, 6, 10, 12, and 14 of the user system connector to GND firmly on the PCB. These pins are used as electrical GND and to monitor the connection of the user system connector. Note the pin assignments of the user system connector.

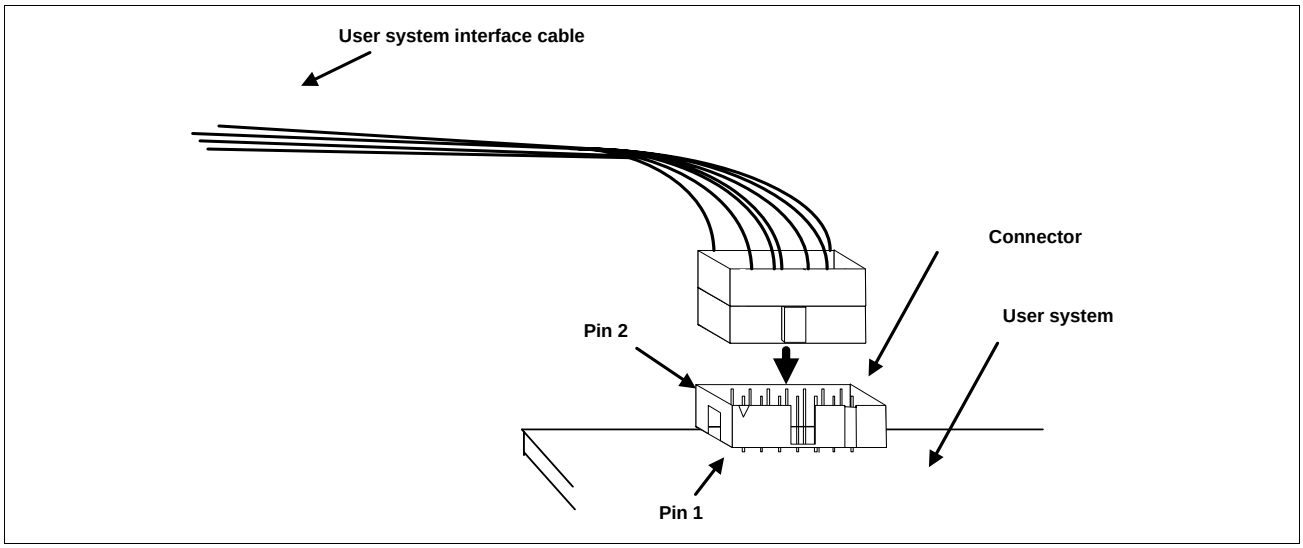
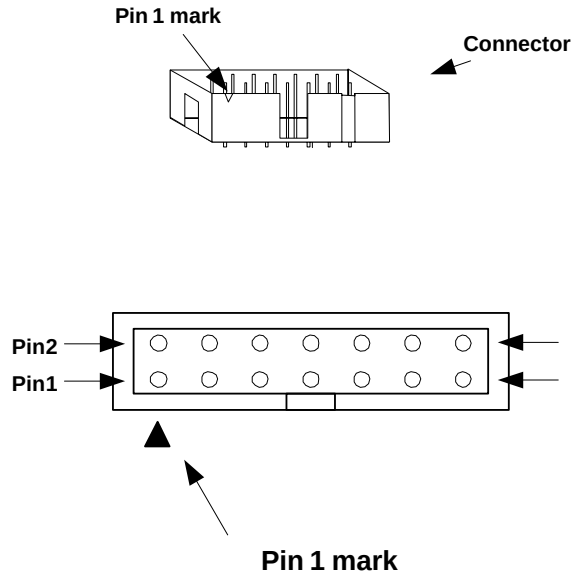


Figure 2.1 Connecting the User System Interface Cable to the User System

Notes: 1. Do not place any components within 3 mm of the connector.
2. When using the E8 emulator as a programmer, connect the E8 emulator to the user system in the same way.

Section 3 Pin Assignments of the E8 Connector

Figure 3.1 shows the pin assignments of the connector.



Pin NO.	**M16C/62P and M16C/6N** MCU signals
1	P65(SCLK)
2	Vss
3	CNVss
4	P55(EPM)
5	P67(TxD)
6	Vss
7	P50(CE)
8	Vcc
9	P64(BUSY)
10	Vss
11	P66(RxD)
12	Vss
13	$\overline{\text{RESET}}$
14	Vss

Figure 3.1 Pin Assignments of the E8 Connector

Note:
Pin 14 is used to check for a connection between the E8 and the user system, and is not directly connected to Vss inside the E8. Make sure that pin 14 and other pins 2, 6, 10, and 12 are connected to Vss.

Section 4 Example of E8 Connection

The connecting examples are shown as follows.
When using the emulator as a programmer, the specification of connection between the E8 and the MCUs is the same as shown in Figure 4.1.

(1) In single-power supply and single-chip mode

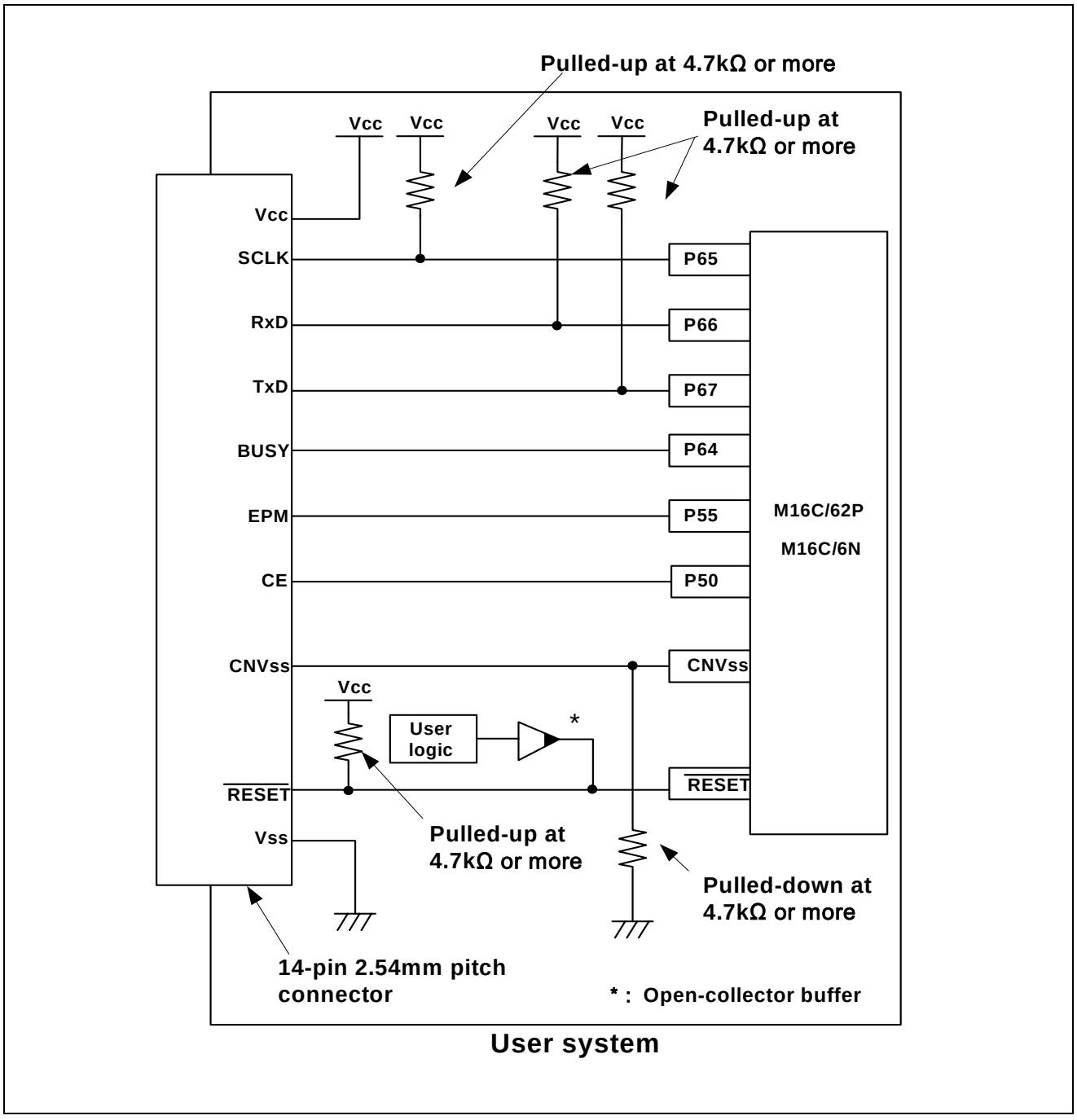


Figure 4.1 Example of E8 Connection (Single-power Supply and Single-chip Mode)

(2) In single-power supply and memory expansion mode

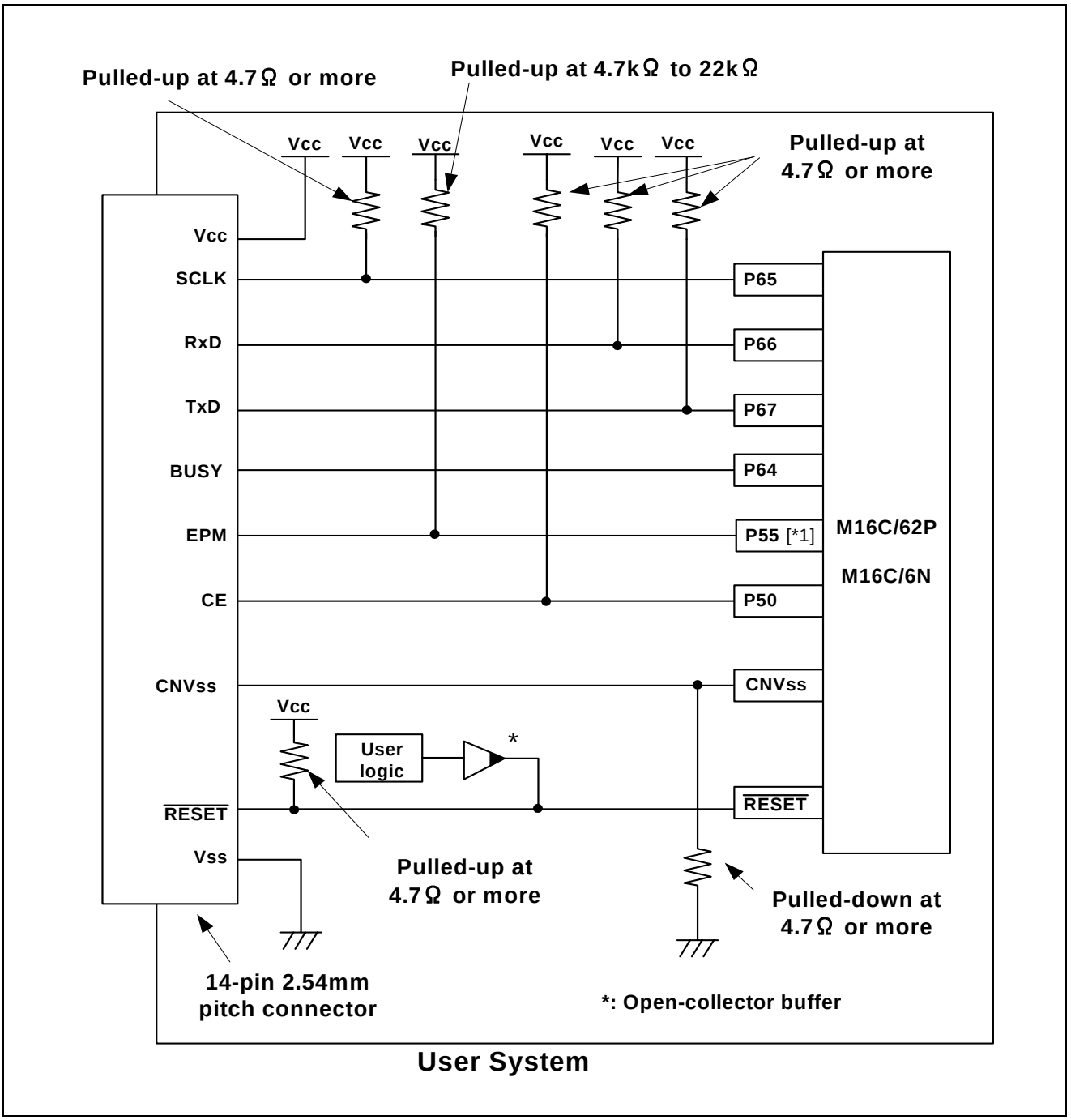


Figure 4.2 Example of E8 Connection (Single-power Supply and Memory Expansion Mode)

[*1]: The $\overline{\text{HOLD}}$ signal cannot be used. Pulled-up P55 on the user system.

(3) In dual-power supply and single-chip mode

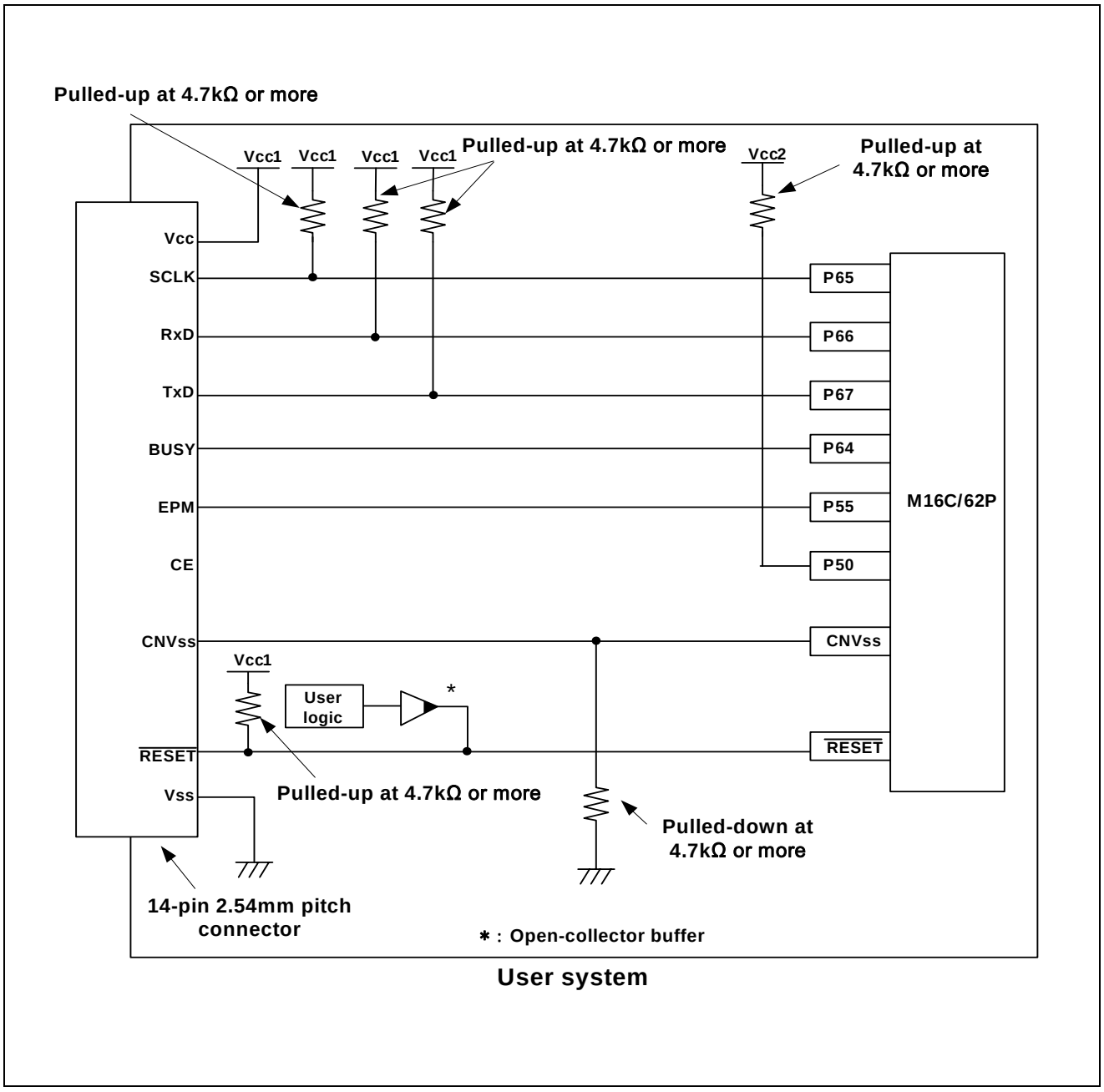


Figure 4.3 Example of E8 Connection (Dual-power Supply and Single-chip Mode)

(4) In dual-power supply and memory expansion mode

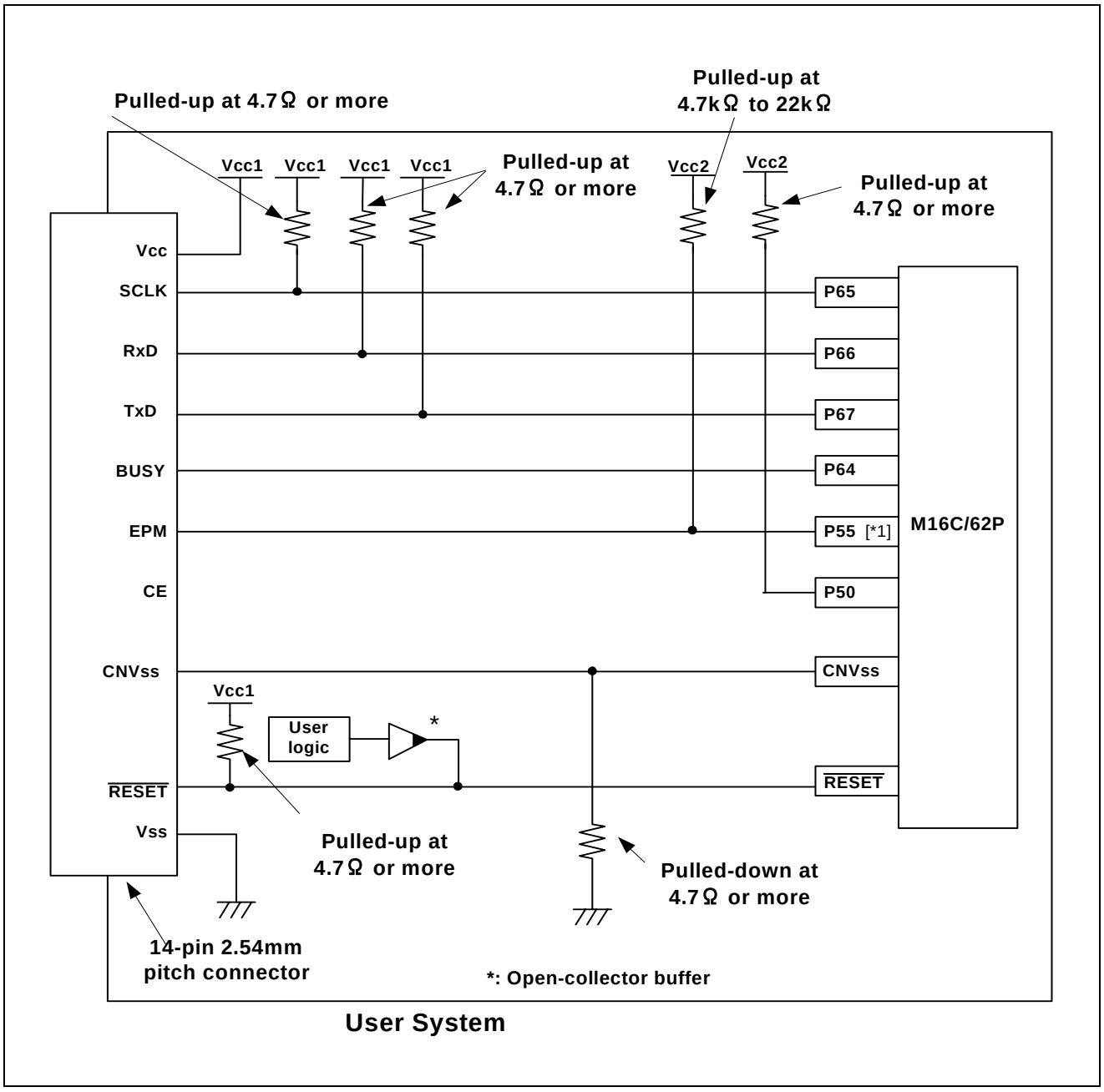


Figure 4.4 Example of E8 Connection (Dual-power Supply and Memory Expansion Mode)

[*1]: The $\overline{\text{HOLD}}$ signal cannot be used. Pulled-up P55 on the user system.

Notes: 1. P64, P65, P66 and P67 pins are used by the E8 emulator.
Connect the E8 emulator to the MCU pins. Pull up MCU pins and connect the E8 emulator to P65, P66 and P67.

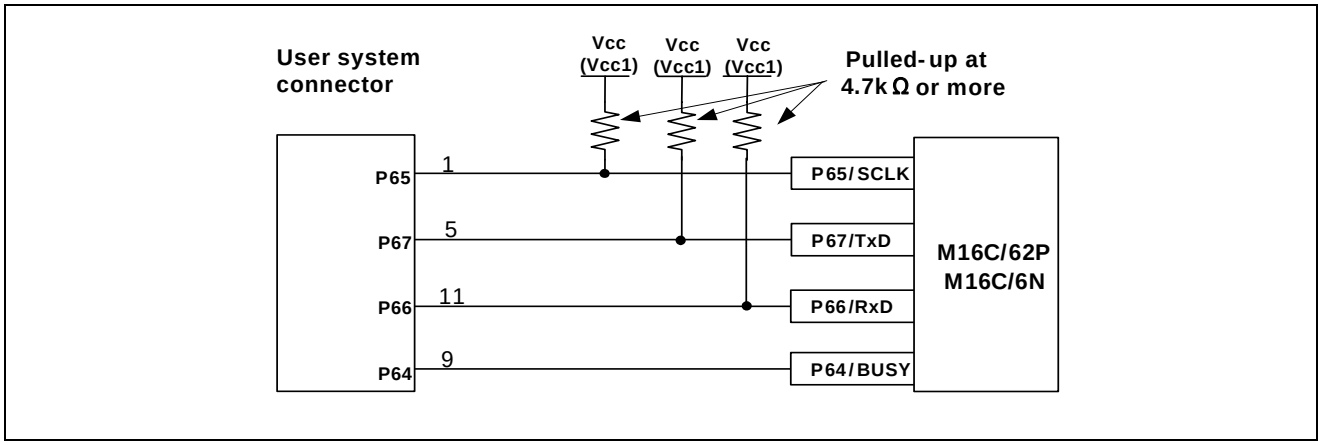


Figure 4.5 Connection of E8 Emulator and MCU

2. The E8 emulator uses the P50 and P55 pins for the MCU control. Connect the E8 emulator to the MCU pins.

(1) In single-power supply and single-chip mode

Figure 4.6 Connection of E8 Emulator and P50 and P55 Pins
(Single-power Supply and Single-chip Mode)

(2) Single-power supply, memory expansion mode

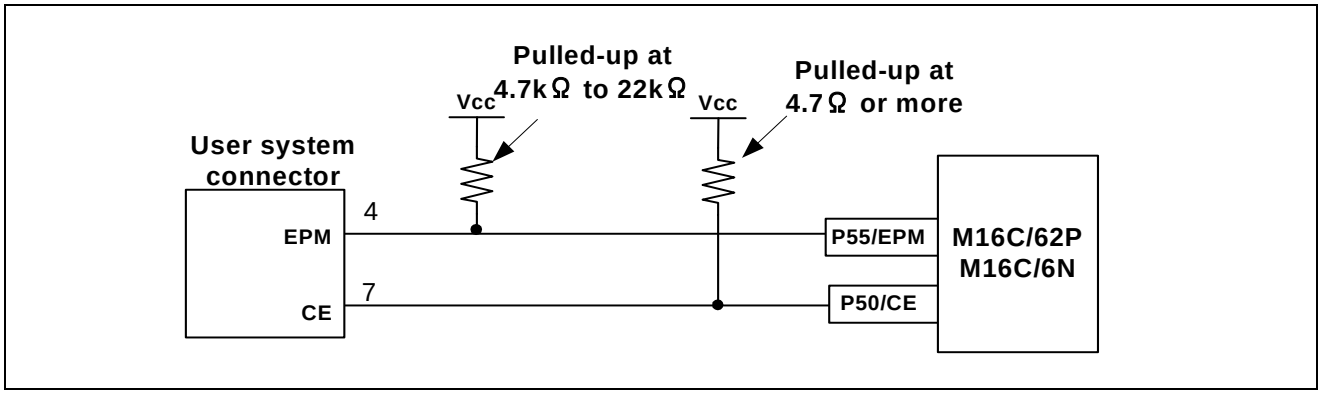


Figure 4.7 Connection of E8 Emulator and P50 and P55 Pins
(Single-power Supply and Memory Expansion Mode)

[*1]: The $\overline{\text{HOLD}}$ signal cannot be used. Pulled-up P55 on the user system.

(3) In dual-power supply and single-chip mode

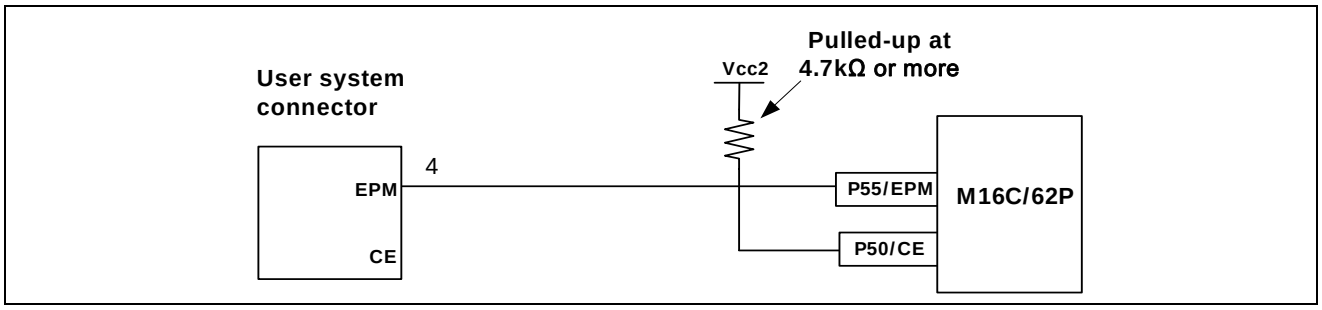


Figure 4.8 Connection of E8 Emulator and P50 and P55 Pins
(Dual-power Supply and Single-chip Mode)

(4) In dual-power supply and memory expansion mode

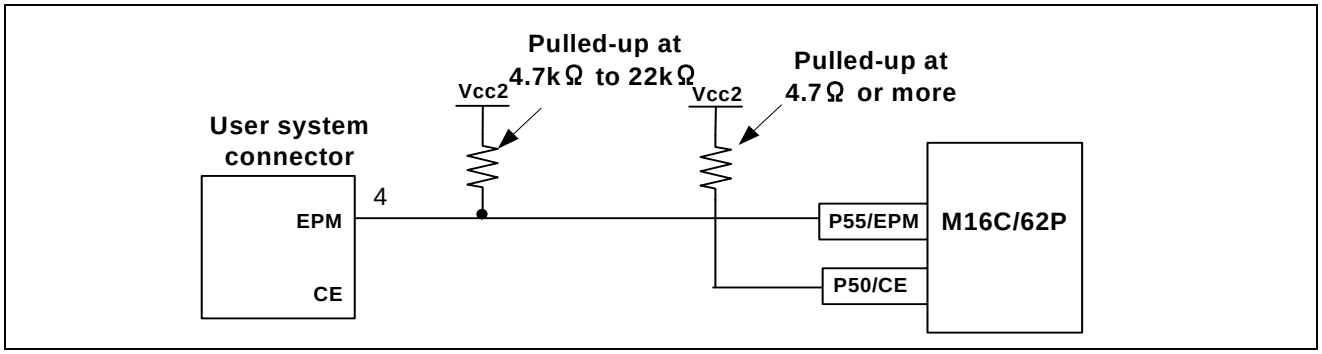


Figure 4.9 Connection of E8 Emulator and P50 and P55 Pins
(Dual-power Supply and Memory Expansion Mode)

[*1]: The $\overline{\text{HOLD}}$ signal cannot be used. Pulled-up P55 on the user system.

3. The E8 emulator uses the CNVss pin for the MCU control. Pulled-down the E8 emulator and the MCU pins to connect the E8 emulator.

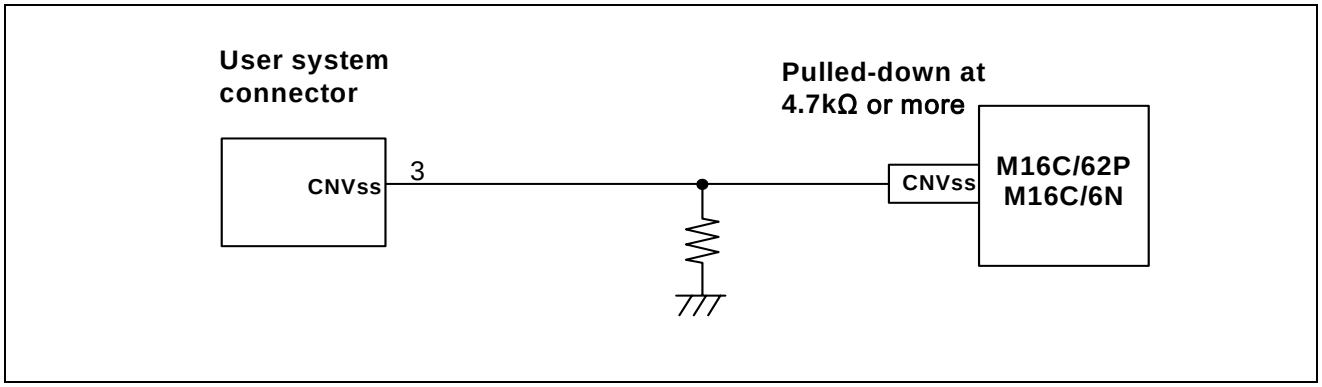


Figure 4.10 Connection of E8 Emulator and CNVss Pin

4. The $\overline{\text{RESET}}$ pin is used by the E8 emulator. Therefore use an open-collector output buffer or a CR reset circuit as the reset circuit of the user system. The recommended pull-up value is 4.7kΩ or more. The MCU can be reset by outputting "L" from the E8 emulator. However, if a reset circuit on the user system is the H-output type reset IC, it cannot be set "L" in the reset circuit on the user system and the E8 emulator will not operate normally.

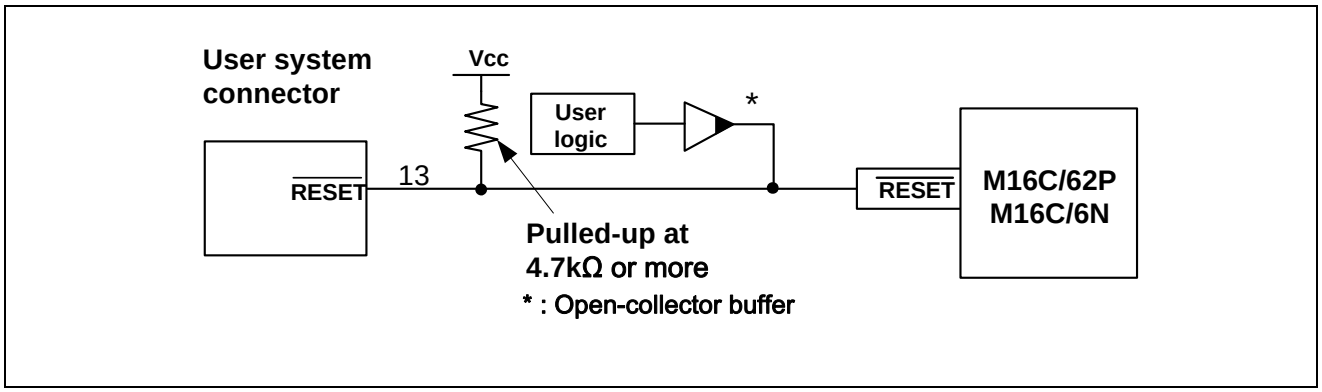
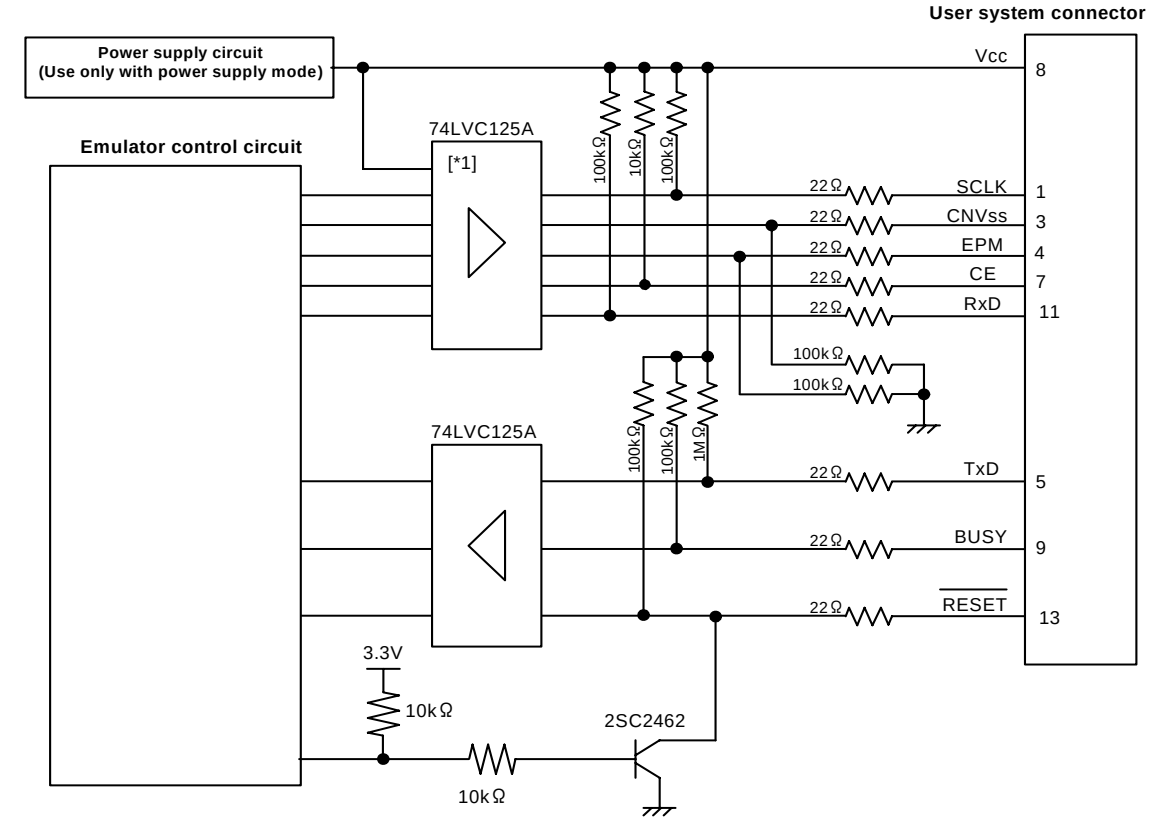


Figure 4.11 Example of a Reset Circuit

5. Connect Vss and Vcc with the Vss and Vcc (Vcc1) of the MCU, respectively.
6. Connect nothing with N.C.
7. The amount of voltage permitted to input to Vcc (Vcc1, Vcc2) must be within the guaranteed range of the MCU.
8. If the $\overline{\text{NMI}}$ pin interrupt are unused, make sure the $\overline{\text{NMI}}$ pin is pulled-up to the Vcc (Vcc1) pin through a resistor.
9. Pin 14 is used to check for a connection between the E8 and the user system, and is not directly connected to Vss inside the E8. Make sure that pin 14 and other pins 2, 6, 10, and 12 are connected to Vss.

10. Figure 4.12 shows the interface circuit in the E8 emulator. Use this figure as a reference when determining the pull-up resistance value.



[*1] Power of the upper 74LVC125A is supplied from Vcc in the user system connector or power supply circuit (with power supply mode).

Figure 4.12 Interface Circuit in the E8 Emulator (Reference)

Section 5 Notes on Using the E8 Emulator

1. Program area for the E8 emulator

   Table 5.1 lists the program area for the E8 emulator.

   Do not change this area. If this area is changed, the E8 emulator will not operate normally. In this case, disconnect to the debugger and then reconnect.

Table 5.1 Program Area for the E8 Emulator

Group	Type Number	ROM Size		RAM Size	Program Area for E8 Emulator		
		Programming Area	Data Area		Vector Area	ROM Area	RAM Area
M16C/62P	M30620FCP	128KB	4KB	10KB	FFFE4h--FFFE7h, FFFE8h--FFFEBh, FFFECh--FFFEFh, FFFF4h--FFFF7h, FFFFCh--FFFFFh	2KB of the programming area [*1]	128 bytes [*1]
	M30621FCP	128KB		10KB			
	M30622F8P	64KB		4KB			
	M30623F8P	64KB		4KB			
	M30624FGP	256KB		20KB			
	M30625FGP	256KB		20KB			
	M30626FHP	384KB		31KB			
	M30626FJP	512KB		31KB			
	M30627FHP	384KB		31KB			
	M30627FJP	512KB		31KB			
	M3062LFGP	256KB		20KB			
	M3062AFC	128KB		10KB			
	M3062CF8	64KB		4KB			
	M3062JFH	384KB		31KB			
M16C/6N4	M306N4FC	128KB		5KB			
	M306N4FG	256KB		10KB			
M16C/6N5	M306N5FC	128KB		5KB			
M16C/6NK	M306NKFH	384KB		31KB			
	M306NKFJ	512KB		31KB			
M16C/6NM	M306NMFH	384KB		31KB			
	M306NMFJ	512KB		31KB			
M16C/6NL	M306NLFH	384KB		31KB			
	M306NLFJ	512KB		31KB			
M16C/6NN	M306NNFH	384KB		31KB			
	M306NNFJ	512KB		31KB			

*1: When starting up the debugger, the [Emulator Setting] dialog box shown in Figure 5.1 is displayed. Specify an area which is not used in the user system.

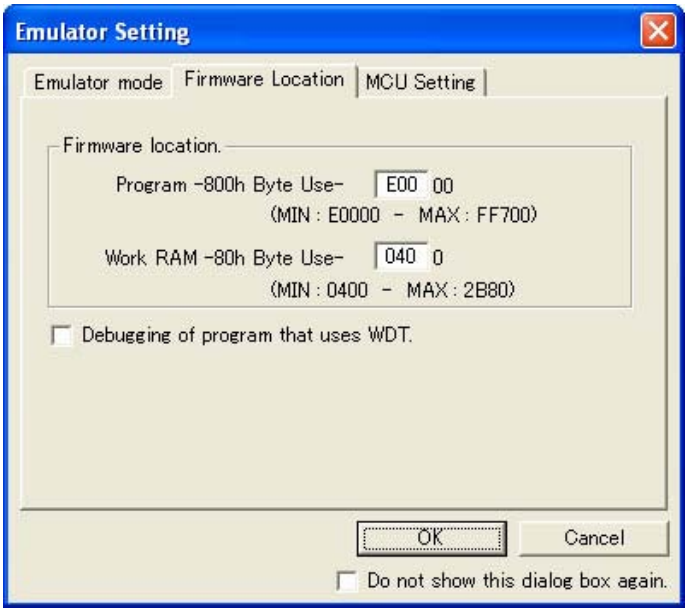


Figure 5.1 [Firmware Location] Tab of the [Emulator Setting] Dialog Box

2. When the emulator system is initiated, it initializes the general registers and part of the control registers as shown in Table 5.2.

Table 5.2 Register Initial Values at Emulator Power-On

Status	Register	Initial Value
E8 Emulator Power-On	PC	Reset vector value in the vector address table
	R0 to R3 (bank 0, 1)	0000h
	A0, A1 (bank 0, 1)	0000h
	FB (bank 0, 1)	0000h
	INTB	0000h
	USP	0000h
	ISP	Work RAM Address for the E8 emulator + 80h *
	SB	0000h
	FLG	0000h

Note: The Work RAM address for the E8 emulator is specified in [Firmware Location] tab of [Emulator Setting] dialog box.

3. The emulator controls the MCUs by using the P50, P55, P64, P65, P66, P67, $\overline{\text{RESET}}$ and CNVss pins.

4. The E8 emulator uses up to 14-byte stack pointer when a user program breaks. Accordingly, reserve the 14-byte addresses for the stack area.

5. SFR used by the program for the E8 emulator

As the SFR listed in Table 5.3 is used by the program for the E8 emulator, do not change a value. Otherwise, the E8 emulator cannot be controlled. Note that UART1 transmit interrupt control register S1TIC and UART1 receive interrupt control register S1RIC always read out the value of using the emulator. Also, they are not initialized by selecting [Debug] -> [Reset CPU] or with the RESET command. If their contents are referred to, a value that has been set in the program for the E8 emulator will be read.

Table 5.3 SFR Used by Program for E8 Emulator

Address	Register	Symbol	Bit	Notes on using the E8 emulator
0009h	Address match interrupt enable register	AIER	All bits	[*1]
0010h - 0012h	Address match interrupt register 0	RMAD0	All bits	[*1]
0014h - 0016h	Address match interrupt register 1	RMAD1	All bits	[*1]
01B8h - 01BAh	Address match interrupt register 2	RMAD2	All bits	[*1]
01BBh	Address match interrupt enable register 2	AIER2	All bits	[*1]
001BC - 01BEh	Address match interrupt register 3	RMAD3	All bits	[*1]
03A8h	UART1 transmit/receive mode register	U1MR	All bits	[*1]
03AAh, 03ABh	UART1 transmit buffer register	U1TB	All bits	[*1]
03ACh	UART1 transmit/receive control register 0	U1C0	All bits	[*1]
03ADh	UART1 transmit/receive control register 1	U1C1	All bits	[*1]
03AEh, 03AFh	UART1 receive buffer register	U1RB	All bits	[*1]
03B0h	UART transmit/receive control register 2	UCON	Bits 1, 3, 4, 5 and 6	[*2]
03ECh	Port P6 register	P6	Bits 4, 5, 6 and 7	[*2]
03EEh	Port P6 direction register	PD6	Bits 4, 5, 6 and 7	[*2]

*1 Do not change the value of the register.

*2 Do not change the value of the bits listed above. When operating this register, change it by a bit operating instruction, etc. in order to avoid changing the value of relevant bits.

6. Interrupts used by the E8 emulator program

The BRK instruction interrupt, address match interrupt, single-step interrupt, and DBC interrupt are used by the E8 emulator program. Therefore, make sure the user program does not use these interrupts. The E8 emulator changes these interrupt vector values to the values to be used by the emulator. It is not a problem if the interrupt vector values are written in the user program.

7. Debugging of the watchdog timer

When debugging the user program using the watchdog timer, select the [Debugging of program that uses WDT.] check box in the [Firmware Location] tab of the [Emulator Setting] dialog box. By selecting this box, the watchdog timer is refreshed during the operation of the program for the E8 emulator. If a memory is accessed by the memory reference or modification, the watchdog timer will be refreshed by the program for the E8 emulator. Note that the operation timing is different from the actual one.

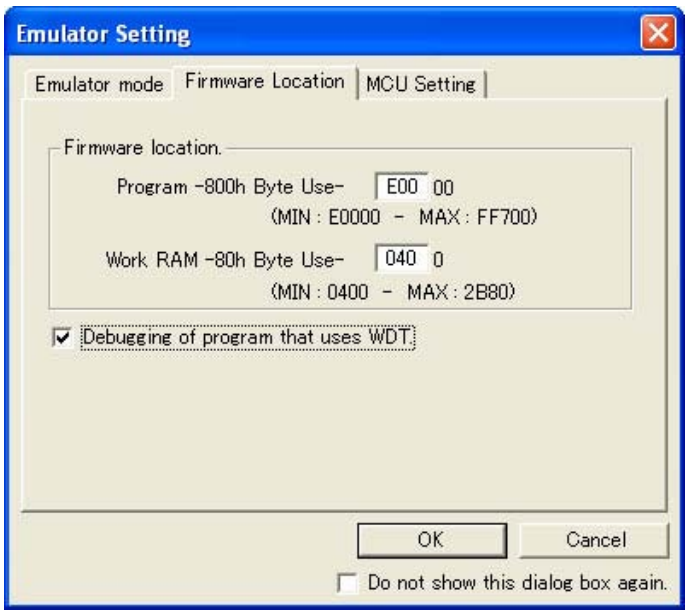


Figure 5.2 [Firmware Location] Tab of the [Emulator Setting] Dialog Box

8. ID code of flash memory

This is the function of the MCUs which prevents the flash memory from reading out by other than the user. The 7 bytes ID code in Table 5.4 written to the flash memory of the MCUs have to match with the ID code displayed in Figure 5.3 [ID Code verification] dialog box at the debugger startup, otherwise the debugger cannot be started up. Note that when the ID code is FFh, FFh, FFh, FFh, FFh, FFh, FFh, ID code is considered not set up. In this case, the ID code is automatically authenticated and [ID Code verification] dialog box is not displayed.

When debugging in [Erase Flash and Connect] mode or [Keep Flash and Connect] mode, the ID code, FFh, FFh, FFh, FFh, FFh, FFh, FFh is written into the ID code area regardless of the contents of the user program. In [Program Flash] mode, the value to be written in the ID code area depends on the contents of the user program.

Table 5.4 ID Code Storage Area of M16C/62P and M16C/6N

Address	Description
FFFDFh	First byte of ID code
FFFE3h	Second byte of ID code
FFFEBh	Third byte of ID code
FFFEFh	Fourth byte of ID code
FFFF3h	Fifth byte of ID code
FFFF7h	Sixth byte of ID code
FFFFBh	Seventh byte of ID code

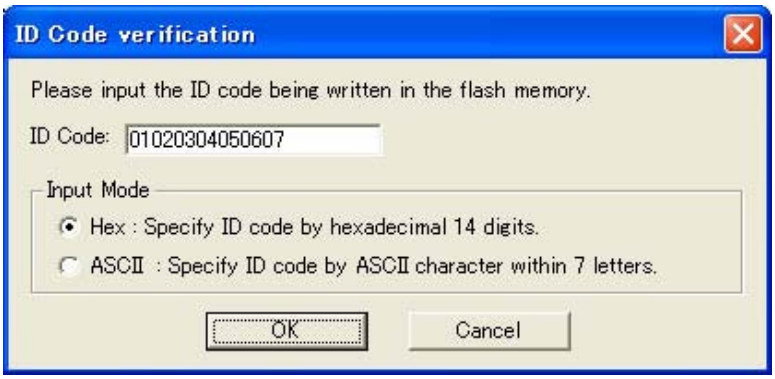


Figure 5.3 [ID Code verification] Dialog Box

[Note on Program Flash mode]

When the ID code is specified by the -ID option of the lmc30, download the MOT file or HEX file. When the X30 file is downloaded, the ID code is not effective. When downloading the X30 file, specify the ID code using an assembler directive command such as ".BYTE". The file to which the ID code specified by the assembler directive command ".ID" is output varies depending on the version of the assembler. For details, refer to the user's manual of the assembler.

9. Operation clock while the user program remains idle

   While the user program remains idle, the E8 emulator program changes the main clock divide-by-N value as it runs.

10. Reset

    The reset vector is used by the E8 emulator program. If the MCU is reset while executing the user program, control is transferred to the E8 emulator program and the user program is made to stop.
    Do not use the hardware reset 2, software reset, watchdog timer reset and oscillation stop detection reset, otherwise the E8 emulator will not operate normally.

11. Memory access during emulation execution

    When referring or modifying the memory contents, the user program is temporarily halted. For this reason, real-time emulation cannot be performed. When the real-time emulation is necessary during the program operation, firstly disable the automatic update in the watch window or fix the display in the memory window so that the memory access will not occur during execution.

12. When the E8 does not supply power to the user system, the E8 emulator consumes the power voltage of the user system from several mA to over 10 mA. This is because the user power supply drives one 74LVC125A to make the communication signal level match the user system power supply voltage.

13. When debugging, the flash memory is frequently re-written by the E8 emulator. Therefore, do not use an MCU that has been used for debugging.
    Also, as the program for the E8 emulator is written into the MCU while debugging, save the contents of the MCU's flash memory that have been used for debugging. Do not use them as the ROM data for products.

14. NMI interrupt

    If NMI interrupts are to be used, be sure to take the necessary measures before executing the user program by, for example, disabling automatic updates of the watch window and freezing the display of the memory window in order to ensure that no memory accesses will occur during user program execution.
    If an NMI interrupt occurs while the user program remains idle or when memory contents are referenced or changed during user program execution, device operation becomes uncontrollable by the E8 emulator.

15. Reserved area

The addresses not specified in the Hardware Manual for the M16C/62P and M16C/6N Groups are reserved area. Do not change the contents. Otherwise, the E8 emulator cannot be controlled.

16. Debugging in the stop mode or wait mode

When debugging in the stop or wait mode, the program cannot be stopped by the E8 emulator. If you attempt to stop the program during the stop or wait mode, the emulator will be uncontrollable.

Do not operate the window until the program stops at the breakpoint by setting the breakpoint at the processing unit where the stop mode or wait mode is cancelled. When using the stop mode or wait mode on a user program, firstly disable the automatic update in the watch window or fix the display in the memory window so that the memory access will not occur during execution.

17. Peripheral I/Os during a break

During a break, although interrupts are not accepted, peripheral I/Os continue to be operated. For example, a timer interrupt is not accepted although counting a timer is continued when a user program is stopped by a break after operating a timer.

18. Exceptional step operation

a) Software-interrupt instruction

STEP operation cannot be performed by continuously executing the internal processing of instructions (undefined, overflow, BRK, and INT) which generates a software interrupt.

<Example> INT instruction

```
            NOP
            NOP
            INT  #3
            NOP          ↘ Passes through if the STEP operation is carried out.
            JMP MAIN
INT_3:
    NOP     ←—— The address at which the program should be stopped.
    NOP
    NOP
    REIT
```

b) INT instruction

Debugging of the program using the INT instruction should be used with the GO command by setting a PC break for the internal processing of the INT instruction.

<Example>

```
            NOP
            INT  #3                    ↓ Execution with the GO command
            NOP
            JMP  MAIN
INT_3:
NOP  Break  ←——
NOP
REIT
```

19. “Go to cursor” function

The "Go to cursor" function is realized by using an address match break. Therefore, when you execute the Go to cursor" command, all the address match breaks you set become invalid, while all the PC breaks remain valid.

20. Note on PC break point

When downloading a user program after changing it, the address setting of a PC break may not be corrected normally depending on the changes. After downloading a user program, please check the setting of a PC break by event point window and reset it.

21. Note on debugging in CPU rewrite mode

When debugging in CPU rewrite mode, do not rewrite the CPU’s block 0 area (addresses FF000h to FFFFFh) and block containing the program for the E8 emulator. If these areas are rewritten, the E8 emulator will run out of control.

Do not halt the user program after setting the CPU rewrite mode until releasing it. If you do so, the E8 emulator may run out of control. Cancel the automatic renewal in the watch window in advance and select fixing display in the memory window to prevent a memory access from occurring while executing the user program.

To check the data after executing the CPU rewrite mode, halt the program after releasing the CPU rewrite mode and see the memory window etc.

When rewriting the flash memory in the program area, select Menu of the High-performance Embedded Workshop – [Setup] -> [Emulator] -> [System...] to open [Configuration] dialog box. In this dialog box, change [Flash memory synchronization] setting to [Flash memory to PC] and set off the cash of the debugger. In this setting, the flash memory is read whenever a break occurs, which takes some time. Use it with [Disable] setting except when debugging in CPU rewrite mode.

22. Note on lock bits of flash memory

When starting up in [Erase Flash and Connect] mode or [Program Flash] mode, lock bits in all the blocks of the flash memory will be unlocked.

Note that the lock bits of the downloaded blocks will be unlocked after downloading the user program.

23. Notes on rewriting the flash memory

Do not reset the MCUs when rewriting the flash memory. The flash memory is completed to rewrite when the “Flash memory write end” is displayed in the output window of the High-performance Embedded Workshop. If the MCU is reset during rewriting the flash memory, the user program or the program for the E8 emulator may be disrupted

The followings indicate when the flash memory rewrite occurs.

- When downloading the user program
- After the user program starts with setting up PC break on the flash memory
- After the user program starts with canceling PC break on the flash memory
- After the user program starts with rewriting the value of flash memory in the memory window

24. Notes on the E8 emulator power supply

When writing reliability required program with the E8 emulator under the mass production, do not use the E8 emulator power supply function. Separately supply appropriate voltage according to the MCU programming to the user system. Since the supplied voltage from the E8 emulator depends on the quality of the USB power supply of the PC, its precision is not guaranteed.

Note that when debugging the system which operates the MCU with dual-power supply, the power cannot be supplied from the E8.

25. Notes on the emulator setup switch

Use the emulator setup switch with the factory setting (upper side “1”).

Section 6 Setup the Debugger

1. Emulator Setting dialog box

[Emulator Setting] dialog box is provided for setting the items that need to be set when the debugger starts up. The contents set from this dialog box other than “Power supply” are also effective the next time the debugger starts.

When starting up the debugger first time after creating the new project work space, [Emulator Setting] dialog box is displayed with the Wizard.

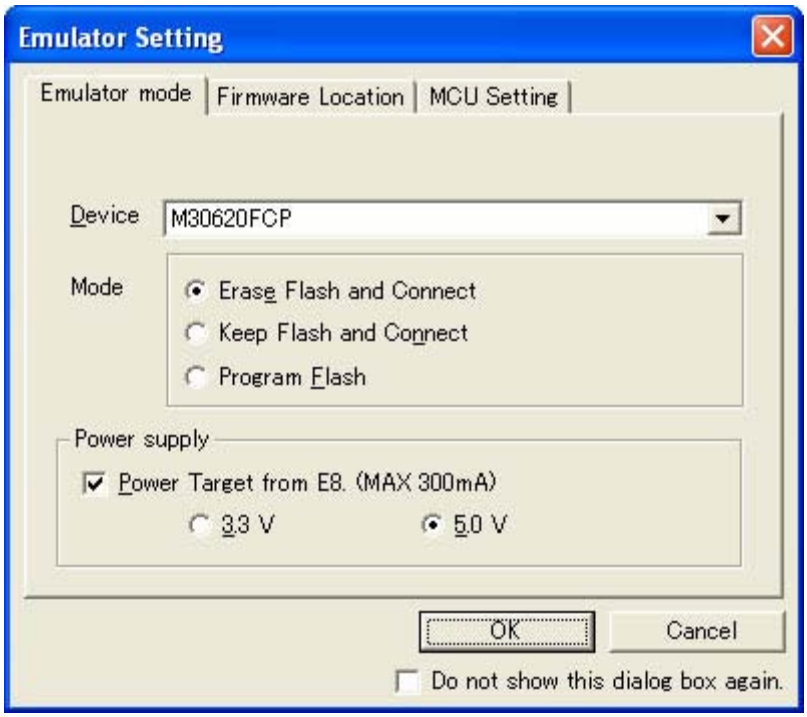


Figure 6.1 [Emulator Setting] Dialog Box.

To keep the [Emulator Setting] dialog box closed the debugger is started next time, check "Do not show this dialog box again." at the bottom of the [Emulator Setting] dialog box. You can open the [Emulator Setting] dialog box using either one of the following methods:

- After the debugger gets started, select Menu - [Setup] -> [Emulator] -> [Emulator Setting (E)...].
- Start Debugger while holding down the Ctrl key.

When you check the "Do not show this dialog box again.", the E8 does not supply power to the user system.

2. Emulator mode Tab

The selection of the device, the specification of the mode, and the setting of the power supply are done in the [Emulator mode] tab of the [Emulator Setting] dialog box.

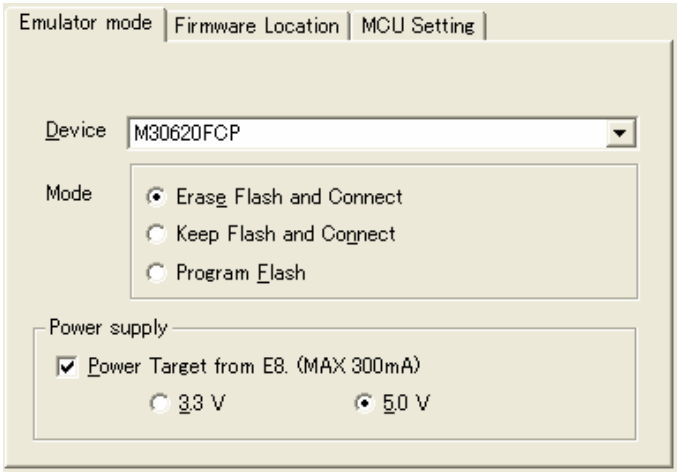


Figure 6.2 [Emulator mode] Tab

[Device]

Select the name of the MCUs to be used from the [Device] drop-down list box.

[Mode]

- Erase Flash and Connect
  When starting up the debugger, erase the flash memory data of the MCUs. The program for the E8 emulator is written at the same time.
- Keep Flash and Connect
  When starting up the debugger, keep the flash memory data of the MCUs. Note that the area for the E8 emulator program and the E8 emulator used vector area will be changed.
- Program Flash
  Select this mode when using the E8 emulator as a simple programmer. On downloading write only the user program (The program for the E8 emulator is not written). Therefore, the program cannot be debugged in this mode.

[Power supply]

When [Power Target from E8. (MAX 300mA)] is checked, power will be supplied to the user system up to 300mA. Note that when debugging the system which operates the MCU with dual-power supply, the power cannot be supplied from the E8.

3. Firmware Location Tab

For details, see “1. Program area for the E8 emulator” and “7. Debugging of the watchdog timer” of the “Section 5. Notes on Using the E8 Emulator”.

4 MCU Setting Tab

In the [MCU Setting] tab, set the operating condition of MCU used in the user system.

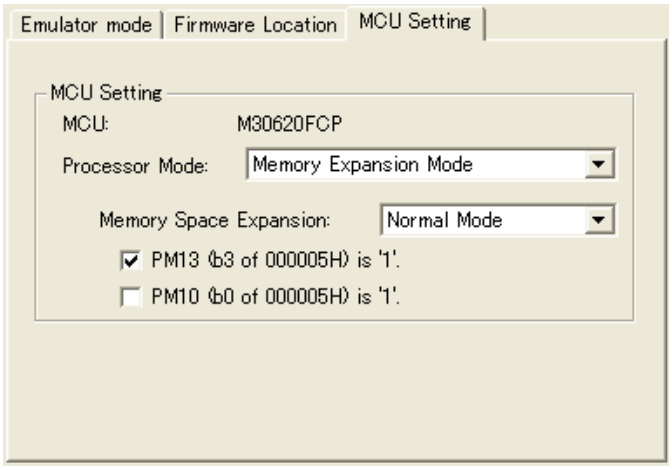


Figure 6.3 [MCU Setting] Dialog Box

Specify the processor mode

Specify the processor mode according to the user system. One of the followings can be specified.

- Single-Chip Mode
- Memory Expansion Mode

Memory Expansion Mode

When the memory expansion mode is selected, specify whether you use the memory space expansion function or not. When using the memory space expansion function, select "4MB Mode", and when not using, select "Normal Mode".

PM13 (bit 3 of 000005H) is "1"

Specify whether you set the PM13 (3rd bit of processor mode register 1). When using your system with the setting that PM13 is 1, check this option.

PM10 (bit 0 of 000005H) is "1"

Specify whether you set the PM10 (0th bit of processor mode register 1). When using your system with the setting that PM10 is 1, check this option.

Notes:

The following describes the precautions to be taken when using the emulator in memory expansion mode.

- When the external area cannot be rewritten via normal memory access, software breaks cannot be used in that area.
- No address match break can be used in external memory space.
- The "Go to Cursor" function cannot be used in external memory space.
  If this function is used in external memory space, a program execution state is entered.
- To access a memory space expansion area in the download, the editor window (MIX display or disassembled display mode), the memory or watch window while operating in memory space expansion 4MB mode, be aware that only bank 7 can be accessed. In this case, the data bank offset depends on the offset bits of the data bank register.
- When use the memory space expansion function 4MB mode, execute the command for the memory space expansion function 4MB mode to access each bank.

Section 7 Command for Memory Space Expansion Function 4MB Mode

1. Command for Memory Space Expansion Function 4MB Mode

The followings show the command for memory space expansion function 4MB mode. These commands can be executed in the command line window.

Command	Description
Memory_Compare_Ext	Compares the memory area (between the start address and the end address) with the memory starting at destination address.
Memory_Display_Ext	Displays memory contents.
Memory_Fill_Ext	Fills an area of memory.
Memory_Find_Ext	Finds a string in a memory range.
Memory_Move_Ext	Moves memory.

2. Details of Command for Memory Space Expansion Function 4MB mode

The followings show the details of the command for memory space expansion function 4MB mode.

Memory_Compare_Ext

Abbreviation: MCE

Description: Compares the memory area (between the start address and the end address) with the memory starting at destination address. This cannot be used during the program execution.

Syntax: MCE <bank> <offsetbit> <start> <end> <destination> [<mode>]

Parameter	**Type**	**Description**
<bank>	Numeric	Bank (0 – 7)
<offsetbit> *	Numeric	Offset bit (0: no offset, 1: offset)
<start>	Numeric	Start address
<end>	Numeric	End address (including this address)
<destination>	Numeric	Destination address
<mode>	Keyword	Format (optional, default = BYTE)
	BYTE	1 bytes
	WORD	2 bytes
	LONG	4 bytes
	DOUBLE	8 bytes

*: When “PM13 is 1” is not selected in the MCU setting dialog box, set the offset bit to “0”

Memory_Display_Ext

Abbreviation: MDE

Description: Displays memory contents. This cannot be used during the program execution.

Syntax: MDE MDE <bank> <offsetbit> <address> [<length>] [<mode>]

Parameter	Type	Description
<bank>	Numeric	Bank (0 – 7)
<offsetbit> *	Numeric	Offset bit (0: no offset, 1: offset)
<address>	Numeric	Start address
<length>	Numeric	Length (optional, default = 0x100 bytes)
<mode>	Keyword	Display format (optional, default = BYTE)
	BYTE	Bytes
	WORD	Words (2 bytes)
	LONG	Long words (4 bytes)
	ASCII	ASCII
	SINGLE	Single-precision floating-point (4 bytes)
	DOUBLE	Double-precision floating-point (8 bytes)

*: When “PM13 is 1” is not selected in the MCU setting dialog box, set the offset bit to “0”

Memory_Fill_Ext

Abbreviation: MFE

Description: Fills an area of memory. This cannot be used during the program execution.

Syntax: MFE <bank> <offsetbit> <start> <end> <data> [<mode>] [<verify>]

Parameter	Type	Description
<bank>	Numeric	Bank (0 – 7)
<offsetbit> *	Numeric	Offset bit (0: no offset, 1: offset)
<start>	Numeric	Start address
<end>	Numeric	End address
<data>	Numeric	Data value
<mode>	Keyword	Data size (optional, default = BYTE)
	BYTE	Byte
	WORD	Word (2 bytes)
	LONG	Long word (4 bytes)
	ASCII	ASCII
	SINGLE	Single-precision floating-point (4 bytes)
	DOUBLE	Double-precision floating-point (8 bytes)
<verify>	Keyword	Verify flag (optional, default = V)
	V	Verify
	N	No verify

*: When “PM13 is 1” is not selected in the MCU setting dialog box, set the offset bit to “0”

Memory_Find_Ext

Abbreviation: MIE

Description: Finds a string in a memory range. This cannot be used during the program execution.

Syntax: MIE <bank> <offsetbit> <start> <end> <string> [<mode>]

Parameter	Type	Description
<bank>	Numeric	Bank (0 – 7)
<offsetbit> *	Numeric	Offset bit (0: no offset, 1: offset)
<start>	Numeric	Start address
<end>	Numeric	End address (including this address)
<string>	Numeric	String to search for
<mode>	Keyword	Format (optional, default = BYTE)
	BYTE	Bytes
	WORD	Words (2 bytes)
	LONG	Long words (4 bytes)
	ASCII	ASCII
	SINGLE	Single-precision floating-point (4 bytes)
	DOUBLE	Double-precision floating-point

*: When “PM13 is 1” is not selected in the MCU setting dialog box, set the offset bit to “0”

Memory_Move_Ext

Abbreviation: MVE

Description: Moves memory. This cannot be used during the program execution.

Syntax: MVE <bank> <offsetbit> <start> <end> <destination> [<verify>] [<mode>]

Parameter	Type	Description
<bank>	Numeric	Bank (0 – 7)
<offsetbit> *	Numeric	Offset bit (0: no offset, 1: offset)
<start>	Numeric	Source start address
<end>	Numeric	Source end address (including this address)
<destination>	Numeric	Destination start address
<verify>	Keyword	Verify flag (optional, default = V)
	V	Verify
	N	No verify
<mode>	Keyword	Format (optional, default = BYTE)
	BYTE	1 byte
	WORD	2 bytes
	LONG	4 bytes
	DOUBLE	8 bytes

*: When “PM13 is 1” is not selected in the MCU setting dialog box, set the offset bit to “0”

Section 8 Applicable Tool Chain and Partner Tools

With the M16C/62P and M16C/6N Groups E8 emulator, you can debug a module created by the inhouse tool chain and third-party products listed in Table 8.1 below.

Table 8.1 Applicable Tool Chain and Partner Tools

Tool chain	M3T-NC30WA V.5.20 Release 01 or later
Partner tools	TASKING M16C C/C++/EC++ Compiler V.2.3r1 or later IAR EWM16C V.2.12 or later

[Precautions on debugging the load modules created in ELF/DWARF2 format]

If the load module was created in ELF/DWARF2 format using TASKING M16C C/C++/EC++ compiler V3.0r1, the precaution described below must be observed when displaying member variables of the base class in the watch window.

<Precaution>

If any class object that has a base class is defined, the following problems may occur:

Case 1: Member variables of the base class cannot directly be referenced from the class object (*1).

Case 2: If the PC value resides in any member function of a derived class, member variables of the base class cannot directly be referenced (*4).

<Solution>

If member variables of the base class need to be referenced in the watch window, follow either method described below.

Case 1: Use indirect references from the class object to refer to member variables of the base class (*2) (*3).

Case 2: Use indirect references from “this” pointer to refer to member variables of the base class (*5) (*6).

<Example program statement>

```
////////////////////////////////////////////////////////////
*.h
    class BaseClass
    {
    public:
        int m_iBase;
    public:
        BaseClass() {
            m_iBase = 0;
        }
        void BaseFunc(void);
    };

    class DerivedClass : public BaseClass
    {
    public:
        int m_iDerive;
    public:
        DerivedClass() {
            m_iDerive   = 0;
        }
        void DerivedFunc(void);
    };
```

*.cpp

```
main()
{
    class DerivedClass ClassObj;
    ClassObj.DerivedFunc();
    return;
}

void BaseClass::BaseFunc(void)
{
    m_iBase = 0x1234;
}

void DerivedClass::DerivedFunc(void)
{
    BaseFunc();
    m_iDerive   = 0x1234;
}
```

//

<Example for registering in the watch window>

//

Case 1: If the PC value resides in the main() function

(1)"ClassObj.m_iBase" : Cannot be referenced (*1)

(2)"ClassObj.__b_BaseClass.m_iBase" : Can be referenced (*2)

(3)"ClassObj"

-"__b_BaseClass"

-"m_iBase" : Can be referenced (*3)

-"m_iDerive"

-: Expansion symbol

Case 2: If the PC value resides in the DerivedClass::DerivedFunc() function

(1)"m_iBase" : Cannot be referenced (*4)

(2)"this->__b_BaseClass.m_iBase" : Can be referenced (*5)

(3)"__b_BaseClass.m_iBase" : Can be referenced (*5)

(4)"this"

-"*"

-"__b_BaseClass"

-"m_iBase" : Can be referenced (*6)

-"m_iDerive"

(5)"__b_BaseClass"

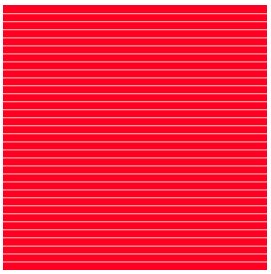
-"m_iBase" : Can be referenced (*6)

//

E8 Emulator
Additional Document for User's Manual
Notes on Connecting the M16C/62P, M16C/6N4, M16C/6N5, M16C/6NK, M16C/6NM, M16C/6NL and M16C/6NN

Publication Date:	December 16, 2005	Rev.1.00
	November 01, 2006	Rev.3.00
Published by:	Sales Strategic Planning Div. Renesas Technology Corp.	
Edited by:	Microcomputer Tool Development Department Renesas Solutions Corp.	

E8 Emulator
Additional Document for User's Manual



Renesas Technology Corp.
2-6-2, Ote-machi, Chiyoda-ku, Tokyo, 100-0004, Japan